IN THE CLAIMS:

1   36.   (Currently Amended)   A method for an original operating system (a

2   host OS) in a mobile device that supports <u>a</u> memory protection mechanism to run

3   another operating system (a guest OS) within the same memory space of said

4   host OS while preserving the current state of said host OS in memory throughout

5   the execution of said guest OS, comprising the steps of:

6   said mobile device running said host OS;

7   said host OS starting said guest OS through a launcher;

8   said launcher <u>going</u> [[go]] through said memory protection mechanism to

9   mark memory blocks currently used by said host OS as protected from <u>said</u>

10   guest OS;

11   said launcher launching said guest OS;

12   said guest OS running, accessing only memory blocks that have not been

13   marked as protected so that said memory blocks marked as protected are

14   preserved throughout the execution of said guest OS;

15   said guest OS <u>finishing</u> <s>finish</s> running through an exit-code;

16   said exit-code <u>restoring</u> <s>restores</s> the state of said host OS by reverting

17   said protected memory blocks; <u>and</u>

18   said host OS resuming its operation.

1   37.   (Currently Amended)   The method of claim 36, wherein said memory

2   protection mechanism is achieved through a memory management unit (MMU)

3   that <u>allows or disallows</u> <s>allow or disallow</s> a program to access particular memory

4   addresses.

1   38.   (Currently Amended)   The method of claim 36, wherein said launcher

2   <u>going</u> [[go]] through said memory protection mechanism to mark memory blocks

3   currently used by said host OS as protected from <u>said</u> guest OS further

4   <u>comprises the</u> <s>comprising</s> steps of:

2

5   moving memory blocks currently used by <u>said</u> host OS whose address

6   range is needed by said guest OS to a free memory location in the <u>mobile</u>

7   device; <u>and</u>

8   marking said free memory blocks as protected;

9   and wherein <u>restoring</u> <s>restores</s> the state of said host OS by reverting said

10   protected memory blocks further <u>comprises the</u> <s>comprising</s> steps of:

11   restoring said free memory blocks to said memory blocks whose address

12   range is needed by said guest OS; <u>and</u>

13   reverting said free memory blocks from protected.

1   39.   (Currently Amended)   The method of claim 36, <u>further comprising</u>

2   <u>starting</u> <s>to start</s> a second guest OS from said guest OS, <s>further</s> <u>wherein starting</u>

3   <u>the second guest OS comprises</u> <s>comprising</s> <u>the</u> steps of:

4   said guest OS acting as a host OS to the second guest OS;

5   said second guest OS acting as a guest OS; <u>and</u>

6   <u>repeating the</u> <s>Repeating</s> steps in claim 36 to start said second guest OS

7   from said guest OS.

1   40.   (Currently Amended)   A method for an original operating system (a

2   host OS) of a mobile device to start another operating system (a guest OS) while

3   keeping the running state and data of said host OS in memory throughout the

4   execution of said guest OS, comprising the steps of:

5   said mobile device running said host OS;

6   said host OS starting said guest OS through a launcher;

7   said launcher moving memory blocks in lower address space that are

8   currently used by said host OS to free memory blocks in upper address space,

9   and eventually preserving current state and data of said host OS to the upper

10   address space through the following steps:

11   for each of said memory blocks in lower address <u>space</u> currently

12   used by said host OS, <u>finding</u> <s>find</s> a free memory block in the upper address

13   space; and

14   moving ~~move~~ said used memory block in lower address space to

15   said free memory block in upper address space;

16   said launcher identifying a memory address location where <u>memory</u>

17   <u>addresses</u> ~~memories~~ above said location contains host OS data and memory

18   addresses below said memory address location <u>are</u> ~~to be~~ free to <u>be used</u> ~~use~~ by

19   said guest OS;

20   said launcher launching said guest OS[[,]] <u>by</u> passing <u>said</u> memory

21   address location as a reduced memory size to said guest OS;

22   said guest OS running in said <u>reduced in size</u> ~~reduce-sized~~ memory space

23   and <u>leaving</u> ~~leave~~ memory space higher than said memory location untouched;

24   said guest OS <u>finishing</u> ~~finish~~ running through an exit-code;

25   said exit-code <u>restoring</u> ~~restores~~ the state and data of said host OS by

26   reverting each of said memory blocks in <u>said</u> lower address <u>space</u> from each of

27   said free memory blocks in <u>said</u> upper address <u>space</u>; <u>and</u>

28   said host OS resuming operation.

1   41.   (Currently Amended)   The method of claim 40, wherein said launcher

2   launching said guest OS[[,]] <u>by</u> passing memory address location as a reduced

3   memory size to said guest OS[[;]] further <u>comprises</u> ~~comprising~~:

4   a memory detection module of said guest OS <u>that</u> uses said <u>reduced in</u>

5   <u>size</u> ~~reduce-sized~~ memory space as the total available memory of the mobile

6   device instead of the real memory size of the mobile device.

1   42.   (Currently Amended)   The method of claim 40, wherein said launcher

2   launching said guest OS[[,]] <u>by</u> passing memory address location as a reduced

3   memory size to said guest OS[[;]] further <u>comprises</u> ~~comprising~~:

4   said launcher modifying <u>a</u> system register in said mobile device to report

5   said reduced memory size as total available <u>memory</u> ~~memories~~ to said guest OS.

1   43.   (Currently Amended)   The method of claim 40, wherein memories of

2   said mobile device are divided into multiple memory banks, and said launcher

3   launching said guest OS[[,]] <u>by</u> passing memory address location as a reduced

4   memory size to said guest OS[[;]] further <u>comprises</u> <s>comprising</s>:

5   <u>generating</u> <s>Generating</s> said reduced memory size by disabling one or

6   more <u>of</u> said memory banks to make them no longer available to said guest OS.

1   44.   (Currently Amended)   The method of claim 40, wherein input output

2   (IO) states[[,]] <u>and</u> registers of said mobile device within said host OS are

3   preserved and later restored into said free memory block in upper address

4   space.

1   45.   (Currently Amended)   A method for an original operating system (a

2   host OS) of a mobile device to start another operating system (a guest OS) within

3   the same memory space of said host OS while keeping the running state of said

4   host OS throughout the execution of said guest OS in place, comprising the

5   steps of:

6   said mobile device running said host OS;

7   said host OS starting said guest OS through a launcher;

8   said launcher launching said guest OS, passing a list of memory

9   addresses of currently used memory of said host OS to a memory device driver

10   in <u>said</u> guest OS during initialization of said guest OS;

11   said memory device driver of said guest OS claiming said list of memory

12   addresses and keeping them from being modified by any other part of said guest

13   OS during the execution of said guest OS;

14   said guest OS running in the same memory space of said host OS, with

15   memories used by said host OS being claimed and protected <u>by</u> <s>under</s> said

16   memory device driver;

17   said guest OS <u>finishing</u> <s>finish</s> running through an exit-code;

18   said exit-code <u>restoring</u> <s>restores</s> the state of said host OS by releasing

19   said list of memory addresses from said device driver; <u>and</u>

20   said host OS resuming operation.

1   46.   (Currently Amended)   The method of claim 45, wherein said

2   list of memory addresses used by host OS includes ~~include~~ memory addresses

3   only in dynamic runtime memory of said host OS and not memories.

1   47.   (Currently Amended)   The method of claim 45, wherein said

2   list of memory addresses used by host OS also includes ~~include~~ the current input

3   output (IO) states and current registers of the mobile device.

1   48.   (Canceled)

1   49.   (Canceled)

1   50.   (Currently Amended)   A method of packaging an image of a guest

2   OS inside a native application of a host OS to allow in-place execution of the said

3   guest OS image in order to reduce memory usage where the format of said

4   native application contains multiple un-continuous data chunks in memory space,

5   comprising the steps of:

6   compiling an image of said guest OS into multiple code segments;

7   appending each of said multiple code segments with a jump table

8   containing multiple jump addresses to be used to point to others of the ~~the other~~

9   code segments;

10   for each of a plurality of inter-segment jump instructions in each of said

11   code segments, linking each of said plurality of inter-segment jump instructions to

12   point to an entry of said jump table;

13   preparing a native application for said host OS with a startup code and

14   multiple data chunks;

15   each of said multiple data chunks wrapping each of said code segments of

16   said guest OS plus each of said jump tables;

17   said host OS launching said guest OS by running said native application;

18   said startup code in said native application looping through <u>entries of</u> each

19   of said jump table ~~entries~~ to fill in the current memory address of each

20   corresponding code <u>segment</u> ~~segments~~;

21   each of said code <u>segments</u> ~~segment~~ executing a jump instruction to

22   invoke codes in <u>the other</u> ~~another~~ code <u>segments</u> ~~segment~~;

23   said executed jump instruction getting <u>a</u> real address of said <u>other</u> ~~another~~

24   code segment from said jump tables;

25   said executed jump instruction successfully jumping to the other code

26   <u>segments</u> ~~segment~~ wrapped by <u>a corresponding</u> ~~said another~~ data chunk; <u>and</u>

27   <u>wherein</u> said image of guest OS <u>executes</u> ~~now executing~~ in place without

28   the need to extract all the code segments from said native application and

29   rearrange them in a continuous memory location.

1   51.   (Currently Amended)   The method of claim 50, wherein said

2   native application in said host OS is a database file with multiple data records or

3   <u>a </u>Palm PDB file with multiple Palm DB records.

1   52.   (Currently Amended)   The method of claim 50, wherein said native

2   application in said host OS is a Palm PDB file with multiple Palm DB records and

3   said host OS is <u>a</u> Palm OS.

1   53.   (Currently Amended)   The method of claim 50, wherein said image of

2   <u>said</u> ~~the~~ guest OS code requires one sequential memory address, said native

3   application of <u>said</u> host OS contains multiple data chunks that <u>have</u> ~~has~~ a

4   maximum chunk size<u>,</u>[[;]] and compiling an image of <u>said</u> guest OS into multiple

5   code segments further <u>comprises</u> ~~comprising~~ the steps of:

6   compiling said image of <u>said</u> guest OS into one code segment with

7   sequential memory <u>addresses</u> ~~address~~;

8   re-arranging instructions in said image of <u>said</u> guest OS to reserve spaces

9   to be used for <u>said</u> jump tables on each of <u>a plurality of</u> ~~said~~ chunk-size

10   boundaries by inserting jump instructions to <u>bypass</u> ~~by-pass~~ those spaces; <u>and</u>

11   splitting said one code segment into multiple code segments on <u>each of</u>

12 <u>said plurality of chunk-size</u> ~~said chunk-size~~ boundaries<u>.</u>[[;]]

1   54.   (Currently Amended)   The method of claim 50, wherein the size of

2 said guest OS image exceeds the maximum size limit of said native application

3 of said host OS, further comprising the steps of:

4   splitting said image of <u>said</u> guest OS into multiple pieces;

5   compiling each <u>of</u> said <u>multiple pieces</u> ~~piece~~ of said image of <u>said</u> guest

6 OS into multiple code segments; <u>and</u>

7   packaging each <u>of</u> said <u>multiple pieces</u> ~~piece~~ of said image of guest OS

8 into a native application of said host OS<u>.</u>[[;]]